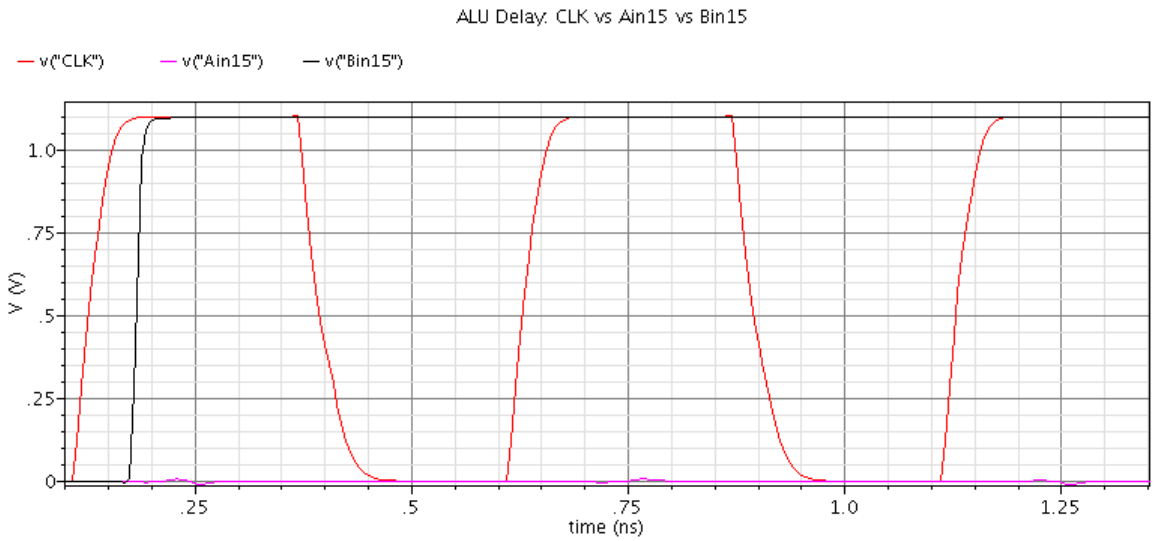
Matthew Ridder,
Doug Etts,
Joseph Fackler

Supplemental Documentation: Team NOR

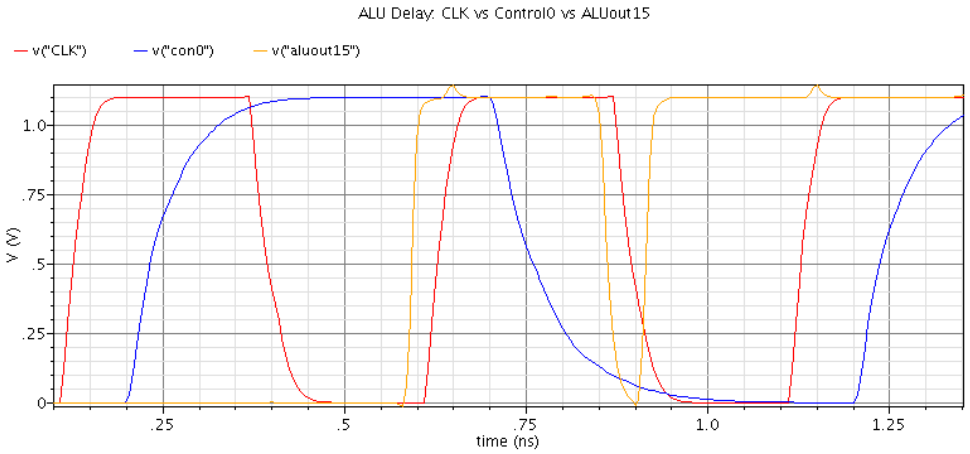
Critical Path through the ALU for the Delay Metric

The critical path of our ALU is triggered when A<15:0> = 0x0000, B<15:0> = 0x8000, and Control<0> switches from 0 to 1. This path is through our Add/Sub block and the subtraction function is performed. Here are a few plots to demonstrate this critical path for our delay metric:



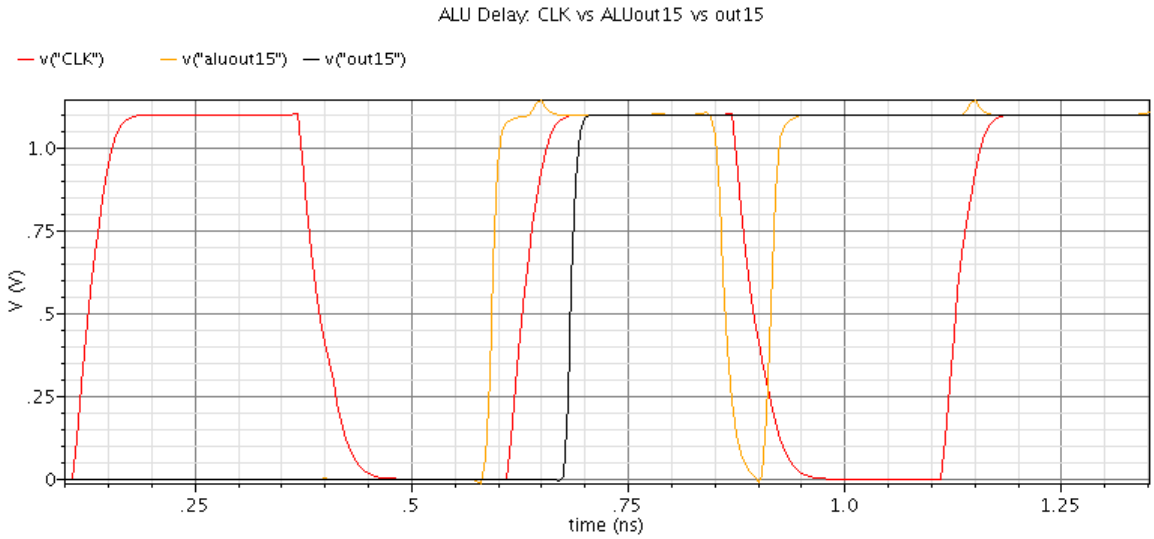
The chart below shows the control bit cycling through logical 1 and 0 values, and the resulting change in ALU output<15> against the CLK signal. The delay of the critical path is shown on this chart, from the rising edge of CLK to when the ALUout<15> settles to its final value.

Note: The vertical axis here is at 0.1 ns. The CLK signal was intentionally delayed to allow registers to be in pre-operation condition before taking a measurement.

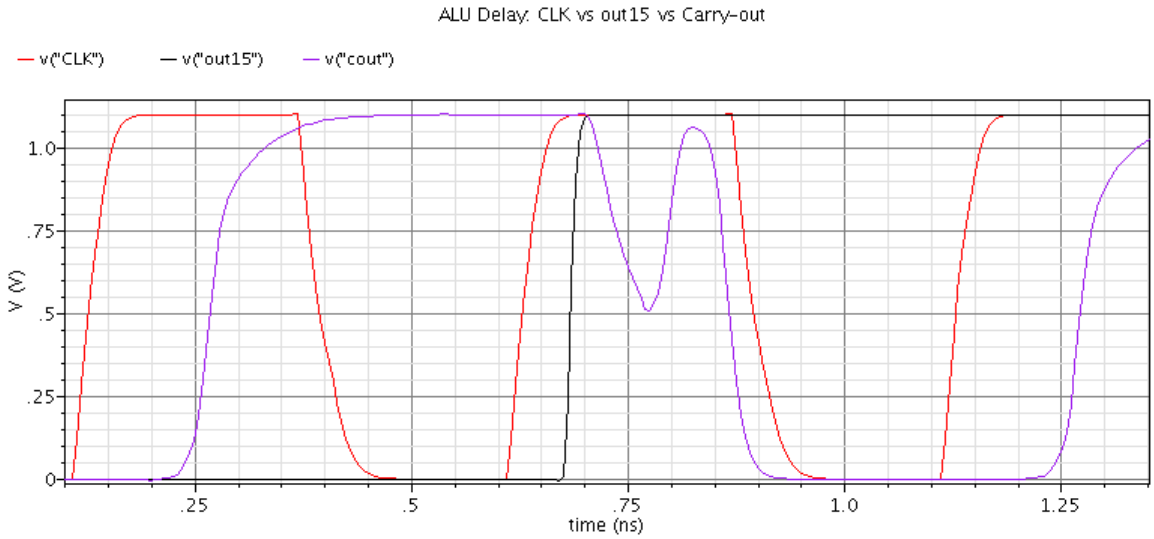


Matthew Ridder,
Doug Etts,
Joseph Fackler

This chart proves that the ALU settled to the correct value before the register at the ALU output latched:



This plot shows the outputs of the DSP block against the CLK signal:



Worst case delay of the ALU through the critical path = **4.641782e-10 s**

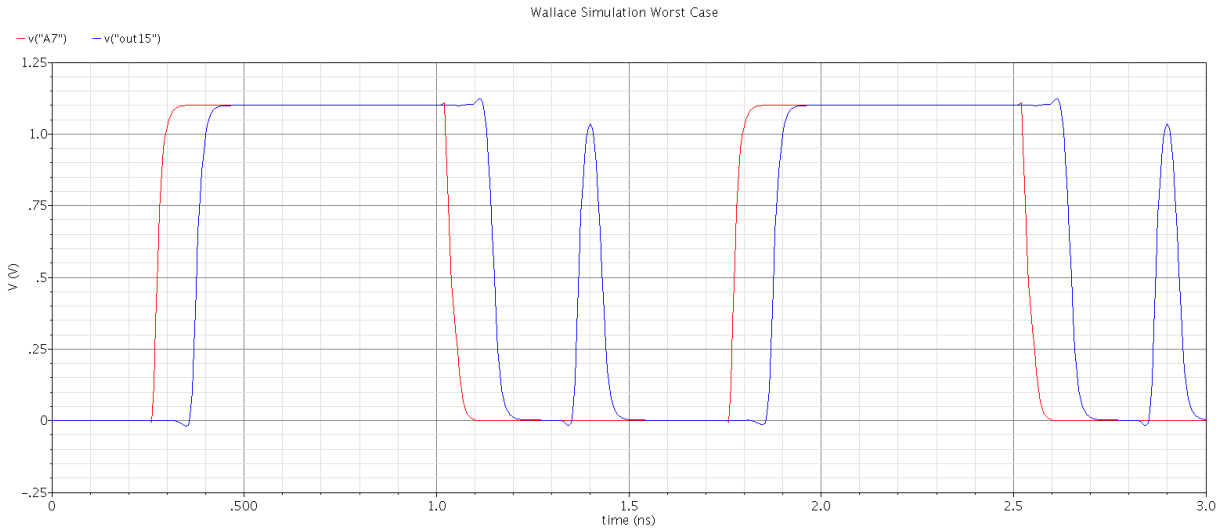
Matthew Ridder,
Doug Etts,
Joseph Fackler

Critical Path through Arbitrary Function Simulation

We chose to build an 8 by 8 Wallace Tree Multiplier for our Arbitrary Function. Here is the functionality summary table that we generated:

Ain(binary)	Ain(decimal)	Bin(Binary)	Bin(decimal)	Output(Binary)	Output(decimal)
11111111	255	11111111	255	1111111000000001	65025
11111111	255	01111111	127	0111111010000001	32385
11111111	255	10111111	191	1011111001000001	48705
11111111	255	00111111	63	0011111011000001	16065

The worst case inputs for our Wallace Tree Multiplier are A<7:0> = 11111111 and B<7:0> = 11111111. Here is a simulation of the worst case delay through the Multiplier:



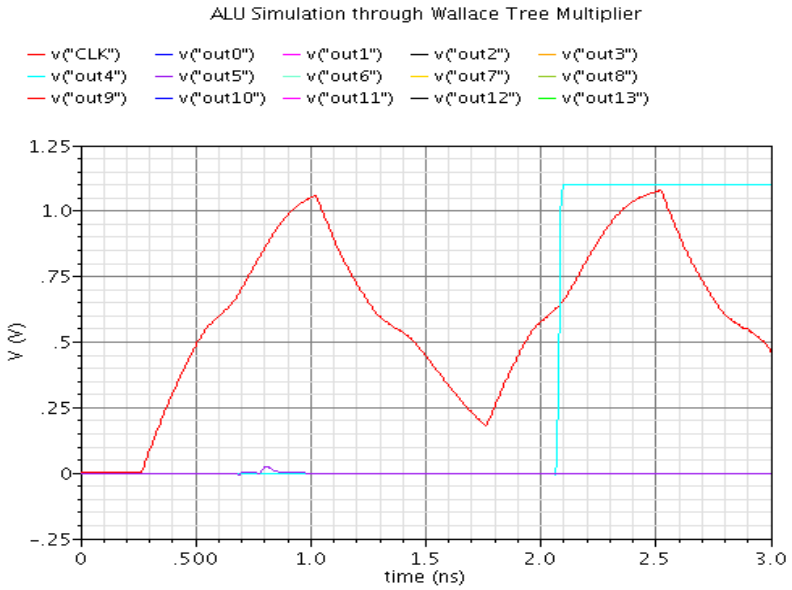
Worst Case Delay = **1.02385e-10 s**

The average power consumption of the Wallace Tree Multiplier was determined by setting A<7:0> to 11111111, and then alternating B<7:0> between 0000000 and 00000001 at 10MHz. The average power consumed was **9.726 mW**.

The area of the Wallace Tree Multiplier is **5.4396e-4 m**.

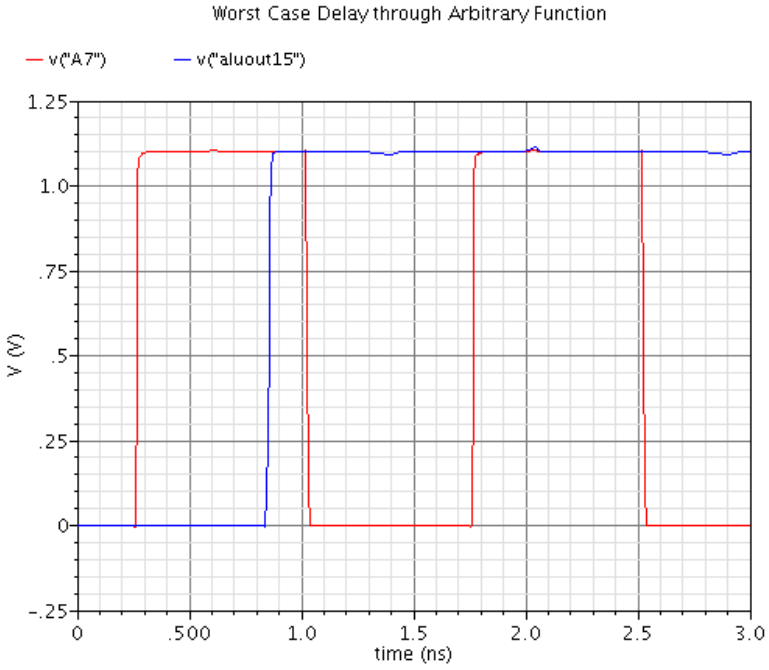
To simulate the critical path of the ALU through our Multiplier, we set A<7:0> = 11111111 and B<7:0> = 11111111. The expected output for out<15:0> = 1111111000000001 or 0xfe01.

Matthew Ridder,
Doug Etts,
Joseph Fackler



As you can see from this plot, the data at out<15:0> was latched at the 2nd clock cycle as expected and it matched the expected result of out<15:0> = 1111111000000001 or 0xfe01.

Here is the plot of the worst case delay of the critical path through our Wallace Tree Multiplier:



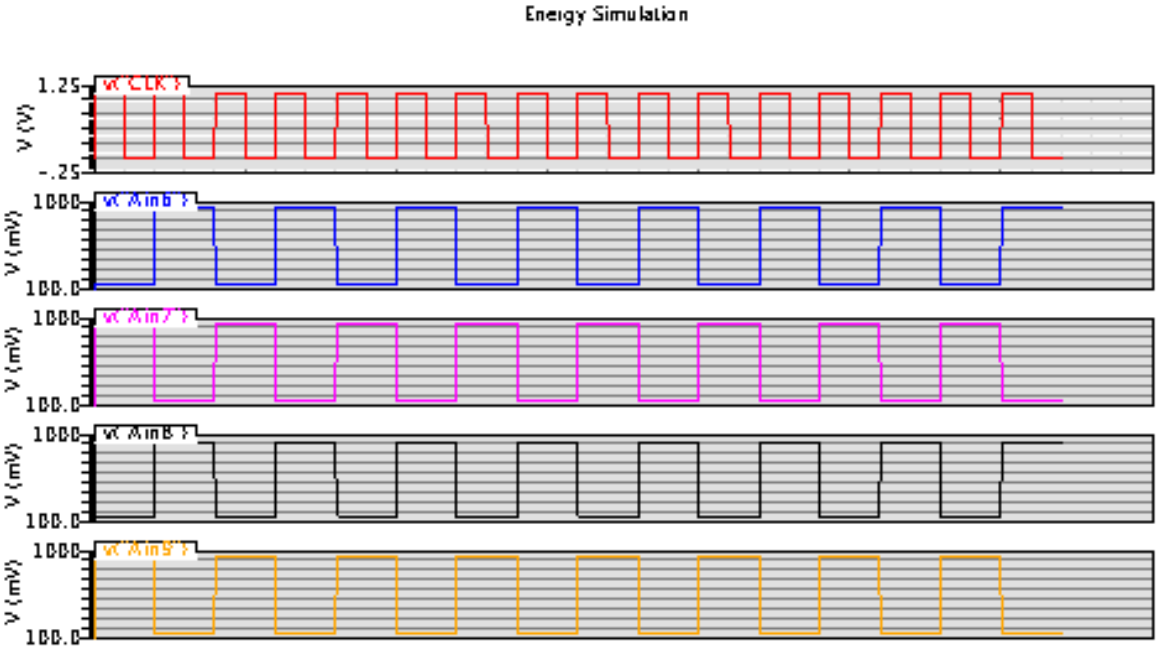
Worst Cade Delay of the ALU through the Multiplier = **5.893310e-10 s**

Matthew Ridder,
Doug Etts,
Joseph Fackler

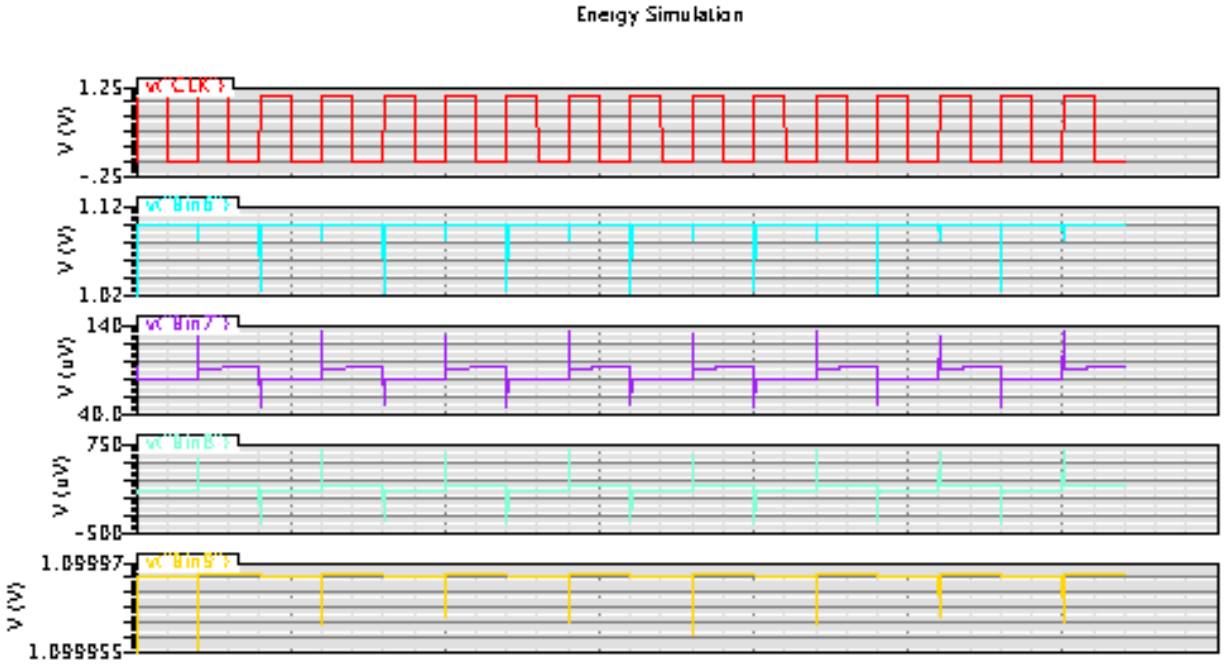
Energy Simulation

These plots demonstrate the setup we used to simulate the energy consumption of our ALU. The simulation time was 0 -> 1600ns in order to count through the control bits twice at 10MHz.

Ain<6:9> : These inputs are switching between 0xAAAA and 0x5555 every cycle.

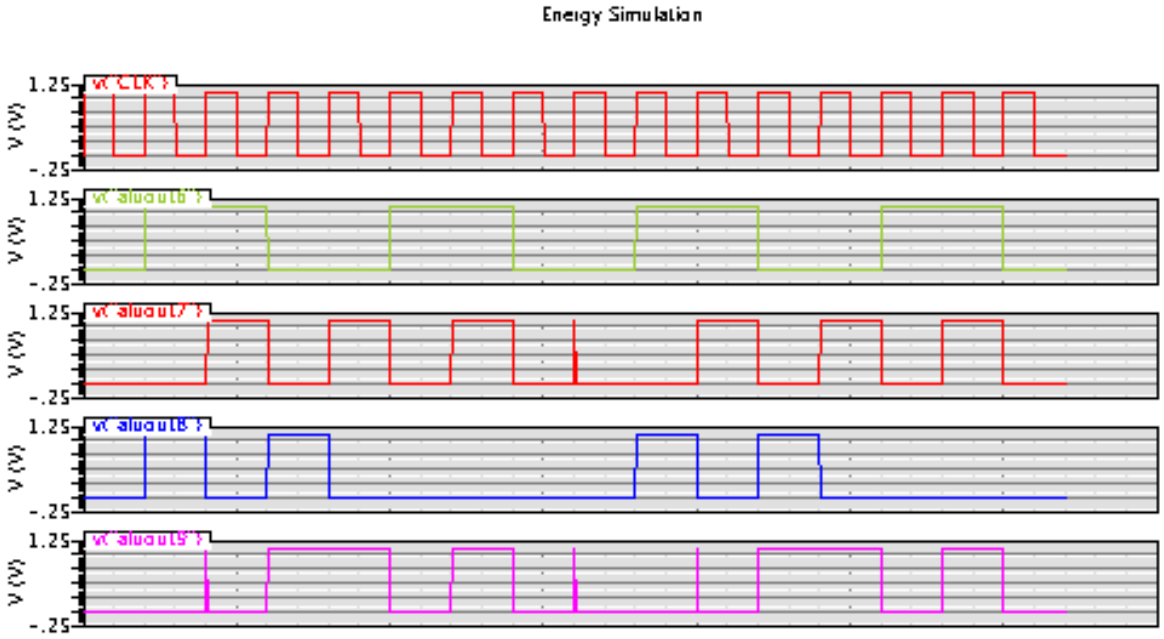


Bin<6:9> : These inputs remain at 0xAA55 for the entire simulation.

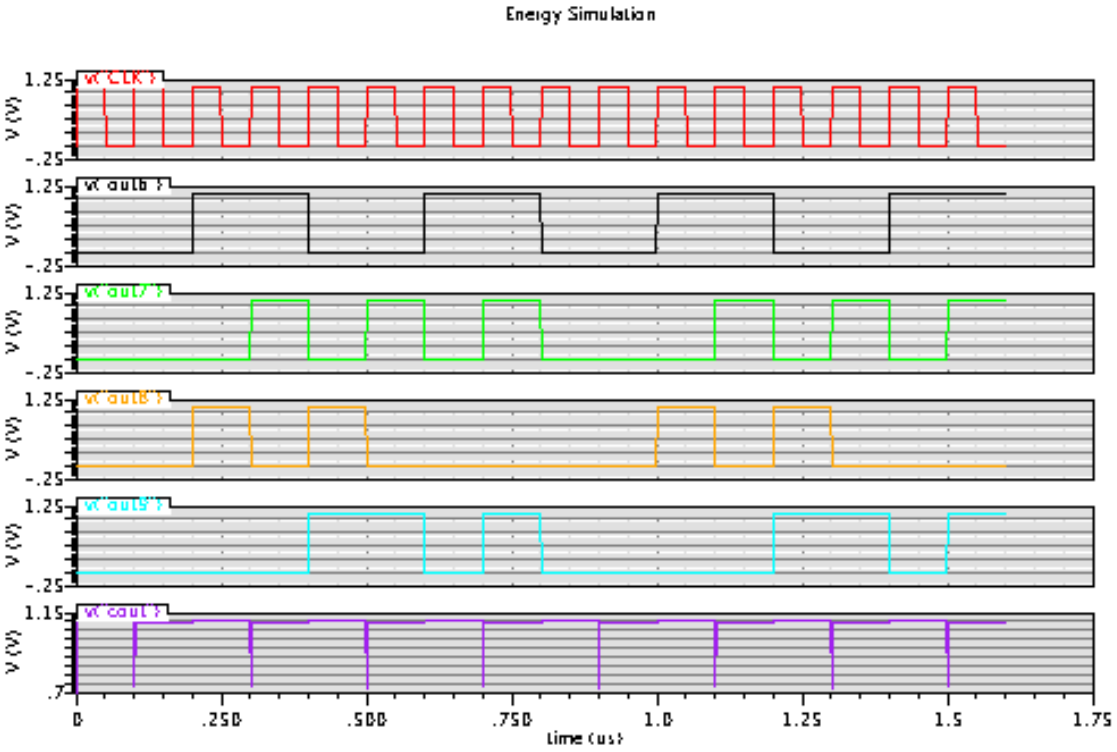


Matthew Ridder,
Doug Etts,
Joseph Fackler

ALUout<6:9> : These are the output signals of the ALU. You can clearly see that the values of the 2nd 8 cycles match the values of the 1st 8 cycles.



Out<6:9> and Cout : Out<6:9> are the ALUout<6:9> signals that get latched by the output register. Cout is the carry out bit from the Add/Sub component.



Matthew Ridder,
Doug Etts,
Joseph Fackler

Average Power/Energy Breakdown

Component	Power (W)	Energy (J)
A Register	0.000682	1.0912E-09
B Register	0.000006	9.6E-12
NOP Register	0.000005	8E-12
Pass A	0.000004	6.4E-12
Add/Sub	0.000081	1.296E-10
Shift	0.000107	1.712E-10
And	0.000012	1.92E-11
Or	0.000011	1.76E-11
Mux	0.00003	4.8E-11
Out Register	0.000005	8E-12
Total	0.000943	1.5088E-09

Area Breakdown

Component	Area (nm)
Register	99360
Pass A	8640
Add/Sub	301050
Shift	91530
And	15840
Or	18720
Mux	298080
Wallace Tree Multiplier	543960
Total (no wallace)	1131300

Operation	Delay (s)
Pass A	3.49E-12
Add	1.07E-10
Sub	3.67E-10
Shift	4.69E-11
And	2.99E-11
Or	2.62E-11

Matthew Ridder,
Doug Etts,
Joseph Fackler

Netlists for Frequently Used Components/Gate Level Netlists

```
// Cell name: TNOR_Inverter
// Function: 1-bit Inverter
// Inputs: in
// Outputs: out
subckt TNOR_Inverter VDD VSS in out
parameters wp=180n wn=90n ln=50n lp=50n mult=1
 MP (out in VDD VDD) PMOS_VTL w=wp l=lp as=100n*wp ad=100n*wp
ps=200n+wp pd=200n+wp m=mult
 MN (out in VSS VSS) NMOS_VTL w=wn l=ln as=100n*wn ad=100n*wn
ps=200n+wn pd=200n+wn m=mult
ends TNOR_Inverter
// End of subcircuit definition

// Cell Name: InvBuffer
// Function: Buffers Input
// Input: A
// Output: out
subckt InvBuffer VDD VSS A out
I0 (VDD VSS A net0) TNOR_Inverter
I1 (VDD VSS net0 out) TNOR_Inverter
ends InvBuffer
// Ends subckt def

// Cell Name: InvCLKBuffer
// Function: Buffers CLK Input
// Input: A
// Output: out
subckt InvCLKBuffer VDD VSS A out
I0 (VDD VSS A net0) TNOR_Inverter wp=720n wn=360n
I1 (VDD VSS net0 out) TNOR_Inverter wp=720n wn=360n
ends InvCLKBuffer
// Ends subckt def

// Cell Name: TNOR_TX
// Function: Transmission Gate
// Input: in A
// Output: out
subckt TNOR_TX VDD VSS in A out
parameters wp=180n wn=90n lp=50n ln=50n mult=1
IO (VDD VSS A Abar) TNOR_Inverter
M0 (in A out out) NMOS_VTL w=wn l=ln as=100n*wn ad=100n*wn ps=200n+wn
pd=200n+wn m=mult
M1 (in Abar out out) PMOS_VTL w=wp l=lp as=100n*wp ad=100n*wp
ps=200n+wp pd=200n+wp m=mult
ends TNOR_TX

// Cell Name: TNOR_NAND2
// 2-Input NAND
```

```
// Inputs: A B
// Output: out
subckt TNOR_NAND2 VDD VSS A B out
parameters wp=180n wn=90n lp=50n ln=50n mult=1
M0 (out A VDD VDD) PMOS_VTL w=wp l=lp as=100n*wp ad=100n*wp ps=200n+wp
pd=200n+wp m=mult
M1 (out B VDD VDD) PMOS_VTL w=wp l=lp as=100n*wp ad=100n*wp ps=200n+wp
pd=200n+wp m=mult
M2 (net5 B VSS VSS) NMOS_VTL w=2*wn l=ln as=100n*2*wn ad=100n*2*wn
ps=200n+2*wn pd=200n+2*wn m=mult
M3 (out A net5 VSS) NMOS_VTL w=2*wn l=ln as=100n*2*wn ad=100n*2*wn
ps=200n+2*wn pd=200n+2*wn m=mult
ends TNOR_NAND2
// end of subckt def

// Cell name: TNOR_AND2
// Function: 2-Input AND
// Inputs: A B
// Outputs: out
subckt TNOR_AND2 VDD VSS A B out
I0 (VDD VSS A B nandout) TNOR_NAND2
I1 (VDD VSS nandout out) TNOR_Inverter
ends TNOR_AND2
// End of subcircuit definition

// Cell name: TNOR_NOR2
// Function: 2-input NOR
//Inputs: A B
//Outputs: out
subckt TNOR_NOR2 VDD VSS A B out
parameters wp=180n wn=90n lp=50n ln=50n mult=1
M1 (out B VSS VSS) NMOS_VTL w=wn l=ln as=100n*wn ad=100n*wn ps=200n+wn
pd=200n+wn m=mult
M0 (out A VSS VSS) NMOS_VTL w=wn l=ln as=100n*wn ad=100n*wn ps=200n+wn
pd=200n+wn m=mult
M2 (out B net0 VDD) PMOS_VTL w=2*wp l=lp as=100n*2*wp ad=100n*2*wp
ps=200n+2*wp pd=200n+2*wp m=mult
M3 (net0 A VDD VDD) PMOS_VTL w=2*wp l=lp as=100n*2*wp ad=100n*2*wp
ps=200n+2*wp pd=200n+2*wp m=mult
ends TNOR_NOR2
// End of subcircuit definition.

// Cell name:TNOR_OR2
//Function: 2-input OR
//Inputs: A B
//Outputs: out
subckt TNOR_OR2 VDD VSS A B out
I0 (VDD VSS A B norout) TNOR_NOR2
I1 (VDD VSS norout out) TNOR_Inverter
ends TNOR_OR2
```

```
// End of subcircuit definition.

// Cell Name: TNOR_XOR
// An XOR gate for the adder
// Inputs: A B
// Outputs: out
subckt TNOR_XOR VDD VSS A B out
parameters wp=180n wn=90n lp=50n ln=50n mult=1
MP0 (net0 A VDD VDD) PMOS_VTL w=wp l=lp as=100n*wp ad=100n*wp
ps=200n+wp pd=200n+wp m=mult
MP1 (out A B VDD) PMOS_VTL w=wp l=lp as=100n*wp ad=100n*wp ps=200n+wp
pd=200n+wp m=mult
MP2 (out B A VDD) PMOS_VTL w=wp l=lp as=100n*wp ad=100n*wp ps=200n+wp
pd=200n+wp m=mult
MN0 (net0 A VSS VSS) NMOS_VTL w=wn l=ln as=100n*wn ad=100n*wn
ps=200n+wn pd=200n+wn m=mult
MN1 (out net0 B VSS) NMOS_VTL w=wn l=ln as=100n*wn ad=100n*wn
ps=200n+wn pd=200n+wn m=mult
MN2 (out B net0 VSS) NMOS_VTL w=wn l=ln as=100n*wn ad=100n*wn
ps=200n+wn pd=200n+wn m=mult
ends TNOR_XOR
// Ends subckt def

// Cell name: TNOR_NAND3
// Function: 3-Input NAND
// Inputs: A B C
// Output: OUT
subckt TNOR_NAND3 VDD VSS A B C OUT
parameters wp=180n wn=90n ln=50n lp=50n mult=1
M2 (net7 C VSS VSS) NMOS_VTL w=3*wn l=ln as=100n*3*wn ad=100n*3*wn
ps=200n+3*wn pd=200n+3*wn m=mult
M1 (OUT A net15 VSS) NMOS_VTL w=3*wn l=ln as=100n*3*wn ad=100n*3*wn
ps=200n+3*wn pd=200n+3*wn m=mult
M0 (net15 B net7 VSS) NMOS_VTL w=3*wn l=ln as=100n*3*wn ad=100n*3*wn
ps=200n+3*wn pd=200n+3*wn m=mult
M5 (OUT C VDD VDD) PMOS_VTL w=wp l=lp as=100n*wp ad=100n*wp ps=200n+wp
pd=200n+wp m=mult
M4 (OUT B VDD VDD) PMOS_VTL w=wp l=lp as=100n*wp ad=100n*wp ps=200n+wp
pd=200n+wp m=mult
M3 (OUT A VDD VDD) PMOS_VTL w=wp l=lp as=100n*wp ad=100n*wp ps=200n+wp
pd=200n+wp m=mult
ends TNOR_NAND3
// End of subcircuit definition

// Cell name: TNOR_AND3
// Function: 3-Input AND
// Inputs: A B C
// Outputs: OUT
subckt TNOR_AND3 VDD VSS A B C OUT
  I1 (VDD VSS A B C nandout) TNOR_NAND3
```

```
  I0 (VDD VSS nandout OUT) TNOR_Inverter
ends TNOR_AND3
// End of subcircuit definition

// Cell name: TNOR_NAND4
// Function: 4-Input NAND
// Inputs: A B C D
// Ouputs: out
// View name: schematic
subckt TNOR_NAND4 VDD VSS A B C D out
parameters wp=180n wn=90n lp=50n ln=50n mult=1
M0 (net0 A VSS VSS) NMOS_VTL w=4*wn l=ln as=100n*4*wn ad=100n*4*wn
ps=200n+4*wn pd=200n+4*wn m=mult
M1 (net1 B net0 VSS) NMOS_VTL w=4*wn l=ln as=100n*4*wn ad=100n*4*wn
ps=200n+4*wn pd=200n+4*wn m=mult
M2 (net2 C net1 VSS) NMOS_VTL w=4*wn l=ln as=100n*4*wn ad=100n*4*wn
ps=200n+4*wn pd=200n+4*wn m=mult
M3 (out D net2 VSS) NMOS_VTL w=4*wn l=ln as=100n*4*wn ad=100n*4*wn
ps=200n+4*wn pd=200n+4*wn m=mult
M4 (out A VDD VDD) PMOS_VTL w=wp l=lp as=100n*wp ad=100n*wp ps=200n+wp
pd=200n+wp m=mult
M5 (out B VDD VDD) PMOS_VTL w=wp l=lp as=100n*wp ad=100n*wp ps=200n+wp
pd=200n+wp m=mult
M6 (out C VDD VDD) PMOS_VTL w=wp l=lp as=100n*wp ad=100n*wp ps=200n+wp
pd=200n+wp m=mult
M7 (out D VDD VDD) PMOS_VTL w=wp l=lp as=100n*wp ad=100n*wp ps=200n+wp
pd=200n+wp m=mult
ends TNOR_NAND4
// End of subcircuit definition for NAND

// Cell Name: Decoder
// Function: 3-8 Decoder, makes one line "hot"
// Input: in0 in1 in2
// Output: out0 out1 out2 out3 out4 out5 out6 out7
subckt Decoder VDD VSS in0 in1 in2 out0 out1 out2 out3 out4 out5 out6
out7
I0 (VDD VSS in0 inv0) TNOR_Inverter
I1 (VDD VSS in1 inv1) TNOR_Inverter
I2 (VDD VSS in2 inv2) TNOR_Inverter
I3 (VDD VSS inv0 inv1 inv2 out0) TNOR_AND3
I4 (VDD VSS in0 inv1 inv2 out1) TNOR_AND3
I5 (VDD VSS inv0 in1 inv2 out2) TNOR_AND3
I6 (VDD VSS in0 in1 inv2 out3) TNOR_AND3
I7 (VDD VSS inv0 inv1 in2 out4) TNOR_AND3
I8 (VDD VSS in0 inv1 in2 out5) TNOR_AND3
I9 (VDD VSS inv0 in1 in2 out6) TNOR_AND3
I10 (VDD VSS in0 in1 in2 out7) TNOR_AND3
ends Decoder

// Cell Name: TNOR_MUX2to1
```

```
// Function: 2:1 Multiplexer
// Inputs: A B S
// Output: out
// Relies on TNOR_Inverter and TNOR_NAND2
subckt TNOR_MUX2to1 VDD VSS A B S out
I0 (VDD VSS S net0) TNOR_Inverter
I1 (VDD VSS A net0 net1) TNOR_NAND2
I2 (VDD VSS B S net2) TNOR_NAND2
I3 (VDD VSS net1 net2 out) TNOR_NAND2
ends TNOR_MUX2to1
// End of subckt def

// Cell Name: TNOR_MUX4to1
// Function: 4:1 Multiplexer
// Inputs: A B C D S0 S1
// Output: out
// Relies on TNOR_Inverter, TNOR_NAND3, and TNOR_NAND4
subckt TNOR_MUX4to1 VDD VSS A B C D S0 S1 out
I0 (VDD VSS S0 net0) TNOR_Inverter
I1 (VDD VSS S1 net1) TNOR_Inverter
I2 (VDD VSS A net0 net1 net2) TNOR_NAND3
I3 (VDD VSS B S0 net1 net3) TNOR_NAND3
I4 (VDD VSS C net0 S1 net4) TNOR_NAND3
I5 (VDD VSS D S0 S1 net5) TNOR_NAND3
I6 (VDD VSS net2 net3 net4 net5 out) TNOR_NAND4
ends TNOR_MUX4to1
// Ends subckt def

// Cell name: Full Adder
// Function: Full Adder used in the Wallace Tree Multiplier
// Inputs: A B Cin
// Ouputs: out Cout
subckt TNOR_FA VDD VSS A B Cin S Cout
I0(VDD VSS A B out1) TNOR_XOR
I1(VDD VSS out1 Cin S) TNOR_XOR
I2(VDD VSS Cin out1 out2) TNOR_AND2
I3(VDD VSS A B out3) TNOR_AND2
I4(VDD VSS out2 out3 Cout) TNOR_OR2
ends TNOR_FA
// End of subcircuit definition.

// Cell name: Half Adder
// Function: Half Adder used in the Wallace Tree Multiplier
// Inputs: A B
// Ouputs: S Cout
subckt TNOR_HA VDD VSS A B S Cout
I0(VDD VSS A B S) TNOR_XOR
I1(VDD VSS A B Cout) TNOR_AND2
ends TNOR_HA
// End of subcircuit definition.
```

```
// Cell Name: TNOR_MUX8to1
// 8:1 Multiplexer
// Inputs: A B C D E F G H S0 S1 S2
// Output: out
// Relies on TNOR_MUX2to1 and TNOR_MUX4to1
subckt TNOR_MUX8to1 VDD VSS A B C D E F G H S0 S1 S2 out
I0 (VDD VSS A B C D S0 S1 net0) TNOR_MUX4to1
I1 (VDD VSS E F G H S0 S1 net1) TNOR_MUX4to1
I2 (VDD VSS net0 net1 S2 out) TNOR_MUX2to1
ends TNOR_MUX8to1
// Ends subckt def

// Cell name: TNOR_REG1
// Function: 1-bit Static Register
// Inputs: D CLK
// Outputs: Q
subckt TNOR_REG1 VDD VSS D CLK Q
I0 (VDD VSS D bufin CLK bufin) TNOR_MUX2to1
I1 (VDD VSS CLK CLKB) TNOR_Inverter
I2 (VDD VSS bufin bufout) InvBuffer
I3 (VDD VSS bufout bufin2 CLKB bufin2) TNOR_MUX2to1
I4 (VDD VSS bufin2 Q) InvBuffer
ends TNOR_REG1
// End of subcircuit definition
```

16-bit Register

```
// Cell name: TNOR_REG
// Function: 16-bit Register
// Inputs: D<15:0> CLK
// Outputs: Q<15:0>
subckt TNOR_REG VDD VSS D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 \
       D12 D13 D14 D15 CLK Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 Q9 Q10 \
       Q11 Q12 Q13 Q14 Q15
I0 (VDD VSS D0 CLK Q0) TNOR_REG1
I1 (VDD VSS D1 CLK Q1) TNOR_REG1
I2 (VDD VSS D2 CLK Q2) TNOR_REG1
I3 (VDD VSS D3 CLK Q3) TNOR_REG1
I4 (VDD VSS D4 CLK Q4) TNOR_REG1
I5 (VDD VSS D5 CLK Q5) TNOR_REG1
I6 (VDD VSS D6 CLK Q6) TNOR_REG1
I7 (VDD VSS D7 CLK Q7) TNOR_REG1
I8 (VDD VSS D8 CLK Q8) TNOR_REG1
I9 (VDD VSS D9 CLK Q9) TNOR_REG1
I10 (VDD VSS D10 CLK Q10) TNOR_REG1
I11 (VDD VSS D11 CLK Q11) TNOR_REG1
I12 (VDD VSS D12 CLK Q12) TNOR_REG1
I13 (VDD VSS D13 CLK Q13) TNOR_REG1
I14 (VDD VSS D14 CLK Q14) TNOR_REG1
```

```
I15 (VDD VSS D15 CLK Q15) TNOR_REG1
ends TNOR_REG
// End of subcircuit definition
```

16-bit PASS A

```
// Cell name: TNOR_PASSA
// Function: 16-bit PASS
// Inputs: A<15:0> B<15:0>
// Outputs: out<15:0>
subckt TNOR_PASSA VDD VSS in0 in1 in2 in3 in4 in5 in6 in7 in8 in9 in10
in11 \
       in12 in13 in14 in15 control out0 out1 out2 out3 out4 out5 \
       out6 out7 out8 out9 out10 out11 out12 out13 out14 out15
I0 (VDD VSS in0 control out0) TNOR_TX
I1 (VDD VSS in1 control out1) TNOR_TX
I2 (VDD VSS in2 control out2) TNOR_TX
I3 (VDD VSS in3 control out3) TNOR_TX
I4 (VDD VSS in4 control out4) TNOR_TX
I5 (VDD VSS in5 control out5) TNOR_TX
I6 (VDD VSS in6 control out6) TNOR_TX
I7 (VDD VSS in7 control out7) TNOR_TX
I8 (VDD VSS in8 control out8) TNOR_TX
I9 (VDD VSS in9 control out9) TNOR_TX
I10 (VDD VSS in10 control out10) TNOR_TX
I11 (VDD VSS in11 control out11) TNOR_TX
I12 (VDD VSS in12 control out12) TNOR_TX
I13 (VDD VSS in13 control out13) TNOR_TX
I14 (VDD VSS in14 control out14) TNOR_TX
I15 (VDD VSS in15 control out15) TNOR_TX
ends TNOR_PASSA
// End of subcircuit definition
```

16-bit Adder/Subtractor

```
// Cell Name: TNOR_CStage_MA
// Function: generates Cbar for the Mirror Adder
// Inputs: A B Cin
// Outputs: Cbar
subckt TNOR_CStage_MA VDD VSS A B Cin Cbar
parameters wp=widfac*180n wn=widfac*90n lp=50n ln=50n mult=1 widfac=1

MP0 (net0 A VDD VDD) PMOS_VTL w=12*wp l=lp as=100n*12*wp ad=100n*12*wp
ps=200n+12*wp pd=200n+12*wp m=mult
MP1 (net0 B VDD VDD) PMOS_VTL w=12*wp l=lp as=100n*12*wp ad=100n*12*wp
ps=200n+12*wp pd=200n+12*wp m=mult
MP2 (Cbar Cin net0 VDD) PMOS_VTL w=12*wp l=lp as=100n*12*wp
ad=100n*12*wp ps=200n+12*wp pd=200n+12*wp m=mult
MP3 (net1 A VDD VDD) PMOS_VTL w=4*wp l=lp as=100n*4*wp ad=100n*4*wp
ps=200n+4*wp pd=200n+4*wp m=mult
```

```
MP4 (Cbar B net1 VDD) PMOS_VTL w=4*wp l=lp as=100n*4*wp ad=100n*4*wp
ps=200n+4*wp pd=200n+4*wp m=mult

MN5 (net2 A VSS VSS) NMOS_VTL w=6*wn l=ln as=100n*6*wn ad=100n*6*wn
ps=200n+6*wn pd=200n+6*wn m=mult
MN6 (net2 B VSS VSS) NMOS_VTL w=6*wn l=ln as=100n*6*wn ad=100n*6*wn
ps=200n+6*wn pd=200n+6*wn m=mult
MN7 (Cbar Cin net2 VSS) NMOS_VTL w=6*wn l=ln as=100n*6*wn ad=100n*6*wn
ps=200n+6*wn pd=200n+6*wn m=mult
MN8 (net3 A VSS VSS) NMOS_VTL w=2*wn l=ln as=100n*2*wn ad=100n*2*wn
ps=200n+2*wn pd=200n+2*wn m=mult
MN9 (Cbar B net3 VSS) NMOS_VTL w=2*wn l=ln as=100n*2*wn ad=100n*2*wn
ps=200n+2*wn pd=200n+2*wn m=mult
ends TNOR_CStage_MA
// Ends subckt def

// Cell name: TNOR_SStage_MA
// Function: Generates Sbar for Mirror Adder
// Inputs: A B Cin Cbar
// Outputs: Sbar
subckt TNOR_SStage_MA VDD VSS A B Cin Cbar Sbar
parameters wp=widfac*180n wn=widfac*90n lp=50n ln=50n mult=1 widfac=1

MP0 (net0 A VDD VDD) PMOS_VTL w=4*wp l=lp as=100n*4*wp ad=100n*4*wp
ps=200n+4*wp pd=200n+4*wp m=mult
MP1 (net0 B VDD VDD) PMOS_VTL w=4*wp l=lp as=100n*4*wp ad=100n*4*wp
ps=200n+4*wp pd=200n+4*wp m=mult
MP2 (net0 Cin VDD VDD) PMOS_VTL w=4*wp l=lp as=100n*4*wp ad=100n*4*wp
ps=200n+4*wp pd=200n+4*wp m=mult
MP3 (Sbar Cbar net0 VDD) PMOS_VTL w=4*wp l=lp as=100n*4*wp
ad=100n*4*wp ps=200n+4*wp pd=200n+4*wp m=mult
MP4 (net1 A VDD VDD) PMOS_VTL w=6*wp l=lp as=100n*6*wp ad=100n*6*wp
ps=200n+6*wp pd=200n+6*wp m=mult
MP5 (net2 B net1 VDD) PMOS_VTL w=6*wp l=lp as=100n*6*wp ad=100n*6*wp
ps=200n+6*wp pd=200n+6*wp m=mult
MP6 (Sbar Cin net2 VDD) PMOS_VTL w=6*wp l=lp as=100n*6*wp ad=100n*6*wp
ps=200n+6*wp pd=200n+6*wp m=mult

MN7 (net3 A VSS VSS) NMOS_VTL w=2*wn l=ln as=100n*2*wn ad=100n*2*wn
ps=200n+2*wn pd=200n+2*wn m=mult
MN8 (net3 B VSS VSS) NMOS_VTL w=2*wn l=ln as=100n*2*wn ad=100n*2*wn
ps=200n+2*wn pd=200n+2*wn m=mult
MN9 (net3 Cin VSS VSS) NMOS_VTL w=2*wn l=ln as=100n*2*wn ad=100n*2*wn
ps=200n+2*wn pd=200n+2*wn m=mult
MN10 (Sbar Cbar net3 VSS) NMOS_VTL w=2*wn l=ln as=100n*2*wn
ad=100n*2*wn ps=200n+2*wn pd=200n+2*wn m=mult
MN11 (net4 A VSS VSS) NMOS_VTL w=3*wn l=ln as=100n*3*wn ad=100n*3*wn
ps=200n+3*wn pd=200n+3*wn m=mult
MN12 (net5 B net4 VSS) NMOS_VTL w=3*wn l=ln as=100n*3*wn ad=100n*3*wn
ps=200n+3*wn pd=200n+3*wn m=mult
```

```
MN13 (Sbar Cin net5 VSS) NMOS_VTL w=3*wn l=ln as=100n*3*wn
ad=100n*3*wn ps=200n+3*wn pd=200n+3*wn m=mult
ends TNOR_SStage_MA
// Ends subckt def

// Cell name: TNOR_MA
// Function: Mirror Adder
// Inputs: A B Cin
// Outputs: Cbar Sbar
subckt TNOR_MA VDD VSS A B Cin Sel Sbar Cbar
parameters wcf=1 wsf=1
I0 (VDD VSS B Sel net0) TNOR_XOR
I1 (VDD VSS A net0 Cin Cbar) TNOR_CStage_MA widfac=wcf
I2 (VDD VSS A net0 Cin Cbar Sbar) TNOR_SStage_MA widfac=wsf
ends TNOR_MA
// Ends subckt def

// Cell name: TNOR_2bitMA
// Funtion: 2-bit Mirror Adder
// Inputs: A0 A1 B0 B1 Cin
// Outputs: S0 S1 Cout
subckt TNOR_2bitMA VDD VSS A0 A1 B0 B1 Cin Sel S0 S1 Cout
parameters wcf0=1 wsf0=1 wcf1=1 wsf1=1
I0 (VDD VSS A0 B0 Cin Sel S0bar C0bar) TNOR_MA wcf=wcf0 wsf=wsf0
I1 (VDD VSS S0bar S0) TNOR_Inverter

I2 (VDD VSS A1 A1bar) TNOR_Inverter
I3 (VDD VSS B1 B1bar) TNOR_Inverter
I4 (VDD VSS A1bar B1bar C0bar Sel S1 Cout) TNOR_MA wcf=wcf1 wsf=wsf1
ends TNOR_2bitMA
// Ends subckt def

// Cell name: TNOR_16bitMA
// Function: 16-bit Mirror Adder
// Inputs: A0-A15, B0-B15
// Outputs: S0-S15, Cout
subckt TNOR_16bitMA VDD VSS A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12
A13 A14 A15 B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12 B13 B14 B15 Cin
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 Cout
parameters wfc0=1 wfs0=1 wfc1=1 wfs1=1 wfc2=1 wfs2=1 wfc3=1 wfs3=1
wfc4=1 wfs4=1 wfc5=1 wfs5=1 wfc6=1 wfs6=1 wfc7=1 wfs7=1 wfc8=1 wfs8=1
wfc9=1 wfs9=1 wfc10=1 wfs10=1 wfc11=1 wfs11=1 wfc12=1 wfs12=1 wfc13=1
wfs13=1 wfc14=1 wfs14=1 wfc15=1 wfs15=1
I0 (VDD VSS A0 A1 B0 B1 Cin Cin S0 S1 net0) TNOR_2bitMA wcf0=wfc0
wsf0=wfs0 wcf1=wfc1 wsf1=wfs1
I1 (VDD VSS A2 A3 B2 B3 net0 Cin S2 S3 net1) TNOR_2bitMA wcf0=wfc2
wsf0=wfs2 wcf1=wfc3 wsf1=wfs3
I2 (VDD VSS A4 A5 B4 B5 net1 Cin S4 S5 net2) TNOR_2bitMA wcf0=wfc4
wsf0=wfs4 wcf1=wfc5 wsf1=wfs5
```

```
I3 (VDD VSS A6 A7 B6 B7 net2 Cin S6 S7 net3) TNOR_2bitMA wcf0=wfc6
wsf0=wfs6 wcf1=wfc7 wsf1=wfs7
I4 (VDD VSS A8 A9 B8 B9 net3 Cin S8 S9 net4) TNOR_2bitMA wcf0=wfc8
wsf0=wfs8 wcf1=wfc9 wsf1=wfs9
I5 (VDD VSS A10 A11 B10 B11 net4 Cin S10 S11 net5) TNOR_2bitMA
wcf0=wfc10 wsf0=wfs10 wcf1=wfc11 wsf1=wfs11
I6 (VDD VSS A12 A13 B12 B13 net5 Cin S12 S13 net6) TNOR_2bitMA
wcf0=wfc12 wsf0=wfs12 wcf1=wfc13 wsf1=wfs13
I7 (VDD VSS A14 A15 B14 B15 net6 Cin S14 S15 CoutB) TNOR_2bitMA
wcf0=wfc14 wsf0=wfs14 wcf1=wfc15 wsf1=wfs15
I8 (VDD VSS Cin CoutB Cout) TNOR_XOR
ends TNOR_16bitMA
// Ends subckt def
```

16-bit Shifter

```
// Cell Name: TNOR_Shifter
// Function: 16 bit Barrel Shifter, can perform left and right shifts
// Inputs: A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 B0 B1
D
// Outputs: out0 out1 out2 out3 out4 out5 out6 out7 out8 out9 out10
out11 out12 out13 out14 out15
subckt TNOR_Shifter VDD VSS A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12
A13 A14 A15 B0 B1 D out0 out1 out2 out3 out4 out5 out6 out7 out8 out9
out10 out11 out12 out13 out14 out15

IO (VDD VSS B0 B1 D hot0 hot1 hot2 hot3 hot4 hot5 hot6 hot7) Decoder

I1 (VDD VSS A1 hot0 buf0) TNOR_TX
I2 (VDD VSS A2 hot0 buf1) TNOR_TX
I3 (VDD VSS A3 hot0 buf2) TNOR_TX
I4 (VDD VSS A4 hot0 buf3) TNOR_TX
I5 (VDD VSS A5 hot0 buf4) TNOR_TX
I6 (VDD VSS A6 hot0 buf5) TNOR_TX
I7 (VDD VSS A7 hot0 buf6) TNOR_TX
I8 (VDD VSS A8 hot0 buf7) TNOR_TX
I9 (VDD VSS A9 hot0 buf8) TNOR_TX
I10 (VDD VSS A10 hot0 buf9) TNOR_TX
I11 (VDD VSS A11 hot0 buf10) TNOR_TX
I12 (VDD VSS A12 hot0 buf11) TNOR_TX
I13 (VDD VSS A13 hot0 buf12) TNOR_TX
I14 (VDD VSS A14 hot0 buf13) TNOR_TX
I15 (VDD VSS A15 hot0 buf14) TNOR_TX
I16 (VDD VSS VSS hot0 buf15) TNOR_TX

I17 (VDD VSS A2 hot1 buf0) TNOR_TX
I18 (VDD VSS A3 hot1 buf1) TNOR_TX
I19 (VDD VSS A4 hot1 buf2) TNOR_TX
I20 (VDD VSS A5 hot1 buf3) TNOR_TX
I21 (VDD VSS A6 hot1 buf4) TNOR_TX
```

```
I22 (VDD VSS A7 hot1 buf5) TNOR_TX
I23 (VDD VSS A8 hot1 buf6) TNOR_TX
I24 (VDD VSS A9 hot1 buf7) TNOR_TX
I25 (VDD VSS A10 hot1 buf8) TNOR_TX
I26 (VDD VSS A11 hot1 buf9) TNOR_TX
I27 (VDD VSS A12 hot1 buf10) TNOR_TX
I28 (VDD VSS A13 hot1 buf11) TNOR_TX
I29 (VDD VSS A14 hot1 buf12) TNOR_TX
I30 (VDD VSS A15 hot1 buf13) TNOR_TX
I31 (VDD VSS VSS hot1 buf14) TNOR_TX
I32 (VDD VSS VSS hot1 buf15) TNOR_TX

I33 (VDD VSS A3 hot2 buf0) TNOR_TX
I34 (VDD VSS A4 hot2 buf1) TNOR_TX
I35 (VDD VSS A5 hot2 buf2) TNOR_TX
I36 (VDD VSS A6 hot2 buf3) TNOR_TX
I37 (VDD VSS A7 hot2 buf4) TNOR_TX
I38 (VDD VSS A8 hot2 buf5) TNOR_TX
I39 (VDD VSS A9 hot2 buf6) TNOR_TX
I40 (VDD VSS A10 hot2 buf7) TNOR_TX
I41 (VDD VSS A11 hot2 buf8) TNOR_TX
I42 (VDD VSS A12 hot2 buf9) TNOR_TX
I43 (VDD VSS A13 hot2 buf10) TNOR_TX
I44 (VDD VSS A14 hot2 buf11) TNOR_TX
I45 (VDD VSS A15 hot2 buf12) TNOR_TX
I46 (VDD VSS VSS hot2 buf13) TNOR_TX
I47 (VDD VSS VSS hot2 buf14) TNOR_TX
I48 (VDD VSS VSS hot2 buf15) TNOR_TX

I49 (VDD VSS A4 hot3 buf0) TNOR_TX
I50 (VDD VSS A5 hot3 buf1) TNOR_TX
I51 (VDD VSS A6 hot3 buf2) TNOR_TX
I52 (VDD VSS A7 hot3 buf3) TNOR_TX
I53 (VDD VSS A8 hot3 buf4) TNOR_TX
I54 (VDD VSS A9 hot3 buf5) TNOR_TX
I55 (VDD VSS A10 hot3 buf6) TNOR_TX
I56 (VDD VSS A11 hot3 buf7) TNOR_TX
I57 (VDD VSS A12 hot3 buf8) TNOR_TX
I58 (VDD VSS A13 hot3 buf9) TNOR_TX
I59 (VDD VSS A14 hot3 buf10) TNOR_TX
I60 (VDD VSS A15 hot3 buf11) TNOR_TX
I61 (VDD VSS VSS hot3 buf12) TNOR_TX
I62 (VDD VSS VSS hot3 buf13) TNOR_TX
I63 (VDD VSS VSS hot3 buf14) TNOR_TX
I64 (VDD VSS VSS hot3 buf15) TNOR_TX

I65 (VDD VSS VSS hot4 buf0) TNOR_TX
I66 (VDD VSS A0 hot4 buf1) TNOR_TX
I67 (VDD VSS A1 hot4 buf2) TNOR_TX
I68 (VDD VSS A2 hot4 buf3) TNOR_TX
```

```
I69 (VDD VSS A3 hot4 buf4) TNOR_TX
I70 (VDD VSS A4 hot4 buf5) TNOR_TX
I71 (VDD VSS A5 hot4 buf6) TNOR_TX
I72 (VDD VSS A6 hot4 buf7) TNOR_TX
I73 (VDD VSS A7 hot4 buf8) TNOR_TX
I74 (VDD VSS A8 hot4 buf9) TNOR_TX
I75 (VDD VSS A9 hot4 buf10) TNOR_TX
I76 (VDD VSS A10 hot4 buf11) TNOR_TX
I77 (VDD VSS A11 hot4 buf12) TNOR_TX
I78 (VDD VSS A12 hot4 buf13) TNOR_TX
I79 (VDD VSS A13 hot4 buf14) TNOR_TX
I80 (VDD VSS A14 hot4 buf15) TNOR_TX

I81 (VDD VSS VSS hot5 buf0) TNOR_TX
I82 (VDD VSS VSS hot5 buf1) TNOR_TX
I83 (VDD VSS A0 hot5 buf2) TNOR_TX
I84 (VDD VSS A1 hot5 buf3) TNOR_TX
I85 (VDD VSS A2 hot5 buf4) TNOR_TX
I86 (VDD VSS A3 hot5 buf5) TNOR_TX
I87 (VDD VSS A4 hot5 buf6) TNOR_TX
I88 (VDD VSS A5 hot5 buf7) TNOR_TX
I89 (VDD VSS A6 hot5 buf8) TNOR_TX
I90 (VDD VSS A7 hot5 buf9) TNOR_TX
I91 (VDD VSS A8 hot5 buf10) TNOR_TX
I92 (VDD VSS A9 hot5 buf11) TNOR_TX
I93 (VDD VSS A10 hot5 buf12) TNOR_TX
I94 (VDD VSS A11 hot5 buf13) TNOR_TX
I95 (VDD VSS A12 hot5 buf14) TNOR_TX
I96 (VDD VSS A13 hot5 buf15) TNOR_TX

I97 (VDD VSS VSS hot6 buf0) TNOR_TX
I98 (VDD VSS VSS hot6 buf1) TNOR_TX
I99 (VDD VSS VSS hot6 buf2) TNOR_TX
I100 (VDD VSS A0 hot6 buf3) TNOR_TX
I101 (VDD VSS A1 hot6 buf4) TNOR_TX
I102 (VDD VSS A2 hot6 buf5) TNOR_TX
I103 (VDD VSS A3 hot6 buf6) TNOR_TX
I104 (VDD VSS A4 hot6 buf7) TNOR_TX
I105 (VDD VSS A5 hot6 buf8) TNOR_TX
I106 (VDD VSS A6 hot6 buf9) TNOR_TX
I107 (VDD VSS A7 hot6 buf10) TNOR_TX
I108 (VDD VSS A8 hot6 buf11) TNOR_TX
I109 (VDD VSS A9 hot6 buf12) TNOR_TX
I110 (VDD VSS A10 hot6 buf13) TNOR_TX
I111 (VDD VSS A11 hot6 buf14) TNOR_TX
I112 (VDD VSS A12 hot6 buf15) TNOR_TX

I113 (VDD VSS VSS hot7 buf0) TNOR_TX
I114 (VDD VSS VSS hot7 buf1) TNOR_TX
I115 (VDD VSS VSS hot7 buf2) TNOR_TX
```

```
I116 (VDD VSS VSS hot7 buf3) TNOR_TX
I117 (VDD VSS A0 hot7 buf4) TNOR_TX
I118 (VDD VSS A1 hot7 buf5) TNOR_TX
I119 (VDD VSS A2 hot7 buf6) TNOR_TX
I120 (VDD VSS A3 hot7 buf7) TNOR_TX
I121 (VDD VSS A4 hot7 buf8) TNOR_TX
I122 (VDD VSS A5 hot7 buf9) TNOR_TX
I123 (VDD VSS A6 hot7 buf10) TNOR_TX
I124 (VDD VSS A7 hot7 buf11) TNOR_TX
I125 (VDD VSS A8 hot7 buf12) TNOR_TX
I126 (VDD VSS A9 hot7 buf13) TNOR_TX
I127 (VDD VSS A10 hot7 buf14) TNOR_TX
I128 (VDD VSS A11 hot7 buf15) TNOR_TX

I129 (VDD VSS buf0 out0) InvBuffer
I130 (VDD VSS buf1 out1) InvBuffer
I131 (VDD VSS buf2 out2) InvBuffer
I132 (VDD VSS buf3 out3) InvBuffer
I133 (VDD VSS buf4 out4) InvBuffer
I134 (VDD VSS buf5 out5) InvBuffer
I135 (VDD VSS buf6 out6) InvBuffer
I136 (VDD VSS buf7 out7) InvBuffer
I137 (VDD VSS buf8 out8) InvBuffer
I138 (VDD VSS buf9 out9) InvBuffer
I139 (VDD VSS buf10 out10) InvBuffer
I140 (VDD VSS buf11 out11) InvBuffer
I141 (VDD VSS buf12 out12) InvBuffer
I142 (VDD VSS buf13 out13) InvBuffer
I143 (VDD VSS buf14 out14) InvBuffer
I144 (VDD VSS buf15 out15) InvBuffer
ends TNOR_Shifter
// Ends subckt def
```

16-bit AND

```
// Cell name: TNOR_AND
// Function: 16-bit AND
// Inputs: A<15:0> B<15:0>
// Outputs: out<15:0>
subckt TNOR_AND VDD VSS A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 \
       A12 A13 A14 A15 B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 \
       B11 B12 B13 B14 B15 out0 out1 out2 out3 out4 out5 \
       out6 out7 out8 out9 out10 out11 out12 out13 out14 out15
I0 (VDD VSS A0 B0 out0) TNOR_AND2
I1 (VDD VSS A1 B1 out1) TNOR_AND2
I2 (VDD VSS A2 B2 out2) TNOR_AND2
I3 (VDD VSS A3 B3 out3) TNOR_AND2
I4 (VDD VSS A4 B4 out4) TNOR_AND2
I5 (VDD VSS A5 B5 out5) TNOR_AND2
I6 (VDD VSS A6 B6 out6) TNOR_AND2
```

```
I7 (VDD VSS A7 B7 out7) TNOR_AND2
I8 (VDD VSS A8 B8 out8) TNOR_AND2
I9 (VDD VSS A9 B9 out9) TNOR_AND2
I10 (VDD VSS A10 B10 out10) TNOR_AND2
I11 (VDD VSS A11 B11 out11) TNOR_AND2
I12 (VDD VSS A12 B12 out12) TNOR_AND2
I13 (VDD VSS A13 B13 out13) TNOR_AND2
I14 (VDD VSS A14 B14 out14) TNOR_AND2
I15 (VDD VSS A15 B15 out15) TNOR_AND2
ends TNOR_AND
// End of subcircuit definition
```

16-bit OR

```
// Cell name: TNOR_OR
// Function: 16-bit OR
// Inputs: A<15:0> B<15:0>
// Outputs: out<15:0>
subckt TNOR_OR VDD VSS A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 \
       A12 A13 A14 A15 B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 \
       B11 B12 B13 B14 B15 out0 out1 out2 out3 out4 out5 \
       out6 out7 out8 out9 out10 out11 out12 out13 out14 out15
I0 (VDD VSS A0 B0 out0) TNOR_OR2
I1 (VDD VSS A1 B1 out1) TNOR_OR2
I2 (VDD VSS A2 B2 out2) TNOR_OR2
I3 (VDD VSS A3 B3 out3) TNOR_OR2
I4 (VDD VSS A4 B4 out4) TNOR_OR2
I5 (VDD VSS A5 B5 out5) TNOR_OR2
I6 (VDD VSS A6 B6 out6) TNOR_OR2
I7 (VDD VSS A7 B7 out7) TNOR_OR2
I8 (VDD VSS A8 B8 out8) TNOR_OR2
I9 (VDD VSS A9 B9 out9) TNOR_OR2
I10 (VDD VSS A10 B10 out10) TNOR_OR2
I11 (VDD VSS A11 B11 out11) TNOR_OR2
I12 (VDD VSS A12 B12 out12) TNOR_OR2
I13 (VDD VSS A13 B13 out13) TNOR_OR2
I14 (VDD VSS A14 B14 out14) TNOR_OR2
I15 (VDD VSS A15 B15 out15) TNOR_OR2
ends TNOR_OR
// End of subcircuit definition
```

Matthew Ridder,
Doug Etts,
Joseph Fackler

8 by 8 Wallace Tree Multiplier

```
// Cell name: 8x8 Wallace Tree Multiplier
// Function: Multiplies
// Inputs: X0 X1 X2 X3 X4 X5 X6 X7 Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
// Ouputs: S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 Cout
subckt TNOR_Wallace VDD VSS X0 X1 X2 X3 X4 X5 X6 X7 Y0 Y1 Y2 Y3 Y4 Y5
Y6 Y7 S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 Cout
//Weight = 1
I0 (VDD VSS X0 Y0 x0y0) TNOR_AND2

//Weight = 2
I1 (VDD VSS X1 Y0 x1y0) TNOR_AND2
I2 (VDD VSS X0 Y1 x0y1) TNOR_AND2
//Weight = 4
I3 (VDD VSS X1 Y1 x1y1) TNOR_AND2
I4 (VDD VSS X2 Y0 x2y0) TNOR_AND2
I5 (VDD VSS X0 Y2 x0y2) TNOR_AND2
//Weight = 8
I6 (VDD VSS X1 Y2 x1y2) TNOR_AND2
I7 (VDD VSS X2 Y1 x2y1) TNOR_AND2
I8 (VDD VSS X3 Y0 x3y0) TNOR_AND2
I9 (VDD VSS X0 Y3 x0y3) TNOR_AND2
//Weight = 16
I10 (VDD VSS X2 Y2 x2y2) TNOR_AND2
I11 (VDD VSS X4 Y0 x4y0) TNOR_AND2
I12 (VDD VSS X0 Y4 x0y4) TNOR_AND2
I13 (VDD VSS X1 Y3 x1y3) TNOR_AND2
I14 (VDD VSS X3 Y1 x3y1) TNOR_AND2
//Weight = 32
I15 (VDD VSS X5 Y0 x5y0) TNOR_AND2
I16 (VDD VSS X0 Y5 x0y5) TNOR_AND2
I17 (VDD VSS X4 Y1 x4y1) TNOR_AND2
I18 (VDD VSS X1 Y4 x1y4) TNOR_AND2
I19 (VDD VSS X2 Y3 x2y3) TNOR_AND2
I20 (VDD VSS X3 Y2 x3y2) TNOR_AND2
//Weight = 64
I21 (VDD VSS X6 Y0 x6y0) TNOR_AND2
I22 (VDD VSS X0 Y6 x0y6) TNOR_AND2
I23 (VDD VSS X5 Y1 x5y1) TNOR_AND2
I24 (VDD VSS X1 Y5 x1y5) TNOR_AND2
I25 (VDD VSS X2 Y4 x2y4) TNOR_AND2
I26 (VDD VSS X4 Y2 x4y2) TNOR_AND2
I27 (VDD VSS X3 Y3 x3y3) TNOR_AND2
//Weight = 128
I28 (VDD VSS X7 Y0 x7y0) TNOR_AND2
I29 (VDD VSS X0 Y7 x0y7) TNOR_AND2
I30 (VDD VSS X6 Y1 x6y1) TNOR_AND2
I31 (VDD VSS X1 Y6 x1y6) TNOR_AND2
```

```
I32 (VDD VSS X2 Y5 x2y5) TNOR_AND2
I33 (VDD VSS X5 Y2 x5y2) TNOR_AND2
I34 (VDD VSS X3 Y4 x3y4) TNOR_AND2
I35 (VDD VSS X4 Y3 x4y3) TNOR_AND2
//Weight = 256
I36 (VDD VSS X7 Y1 x7y1) TNOR_AND2
I37 (VDD VSS X1 Y7 x1y7) TNOR_AND2
I38 (VDD VSS X6 Y2 x6y2) TNOR_AND2
I39 (VDD VSS X2 Y6 x2y6) TNOR_AND2
I40 (VDD VSS X5 Y3 x5y3) TNOR_AND2
I41 (VDD VSS X3 Y5 x3y5) TNOR_AND2
I42 (VDD VSS X4 Y4 x4y4) TNOR_AND2
//Weight = 512
I43 (VDD VSS X7 Y2 x7y2) TNOR_AND2
I44 (VDD VSS X2 Y7 x2y7) TNOR_AND2
I45 (VDD VSS X6 Y3 x6y3) TNOR_AND2
I46 (VDD VSS X3 Y6 x3y6) TNOR_AND2
I47 (VDD VSS X4 Y5 x4y5) TNOR_AND2
I48 (VDD VSS X5 Y4 x5y4) TNOR_AND2
//Weight = 1024
I49 (VDD VSS X7 Y3 x7y3) TNOR_AND2
I50 (VDD VSS X3 Y7 x3y7) TNOR_AND2
I51 (VDD VSS X6 Y4 x6y4) TNOR_AND2
I52 (VDD VSS X4 Y6 x4y6) TNOR_AND2
I53 (VDD VSS X5 Y5 x5y5) TNOR_AND2
//Weight = 2048
I54 (VDD VSS X7 Y4 x7y4) TNOR_AND2
I55 (VDD VSS X4 Y7 x4y7) TNOR_AND2
I56 (VDD VSS X6 Y5 x6y5) TNOR_AND2
I57 (VDD VSS X5 Y6 x5y6) TNOR_AND2
//Weight = 4096
I58 (VDD VSS X7 Y5 x7y5) TNOR_AND2
I59 (VDD VSS X5 Y7 x5y7) TNOR_AND2
I60 (VDD VSS X6 Y6 x6y6) TNOR_AND2
//Weight = 8192
I61 (VDD VSS X7 Y6 x7y6) TNOR_AND2
I62 (VDD VSS X6 Y7 x6y7) TNOR_AND2
//Weight = 16384
I63 (VDD VSS X7 Y7 x7y7) TNOR_AND2

// S0 : Pass xoyo Cin = 0 and 0 as B to A0 B0 of mirror adder

//S1: pass x0y1 and x1y0 directly in

//S2: 1 HA and 1 passed directly
I64 (VDD VSS x2y0 x0y2 aS2 aS3) TNOR_HA

//S3:
I65 (VDD VSS x3y0 x0y3 net0 net1) TNOR_HA
// The below produces bs3 and as4
```

```
I66 (VDD VSS x2y1 x1y2 net0 bS3 aS4) TNOR_FA

//S4: 2 FA and 1 Half
I67 (VDD VSS x4y0 x0y4 net1 net2 net3) TNOR_FA
I68 (VDD VSS x3y1 x1y3 x2y2 net4 net5) TNOR_FA
I69 (VDD VSS net2 net4 bS4 aS5) TNOR_HA

//S5: 3 incoming couts; 1 will go to s5
I70 (VDD VSS x5y0 x0y5 net3 net6 net7) TNOR_FA
I71 (VDD VSS x1y4 x4y1 net5 net8 net9) TNOR_FA
I72 (VDD VSS x3y2 x2y3 net6 net10 net11) TNOR_FA
I73 (VDD VSS net10 net8 bS5 aS6) TNOR_HA

//S6:
I74 (VDD VSS x6y0 x0y6 net7 net12 net13) TNOR_FA
I75 (VDD VSS x5y1 x1y5 net9 net14 net15) TNOR_FA
I76 (VDD VSS x4y2 x2y4 net11 net16 net17) TNOR_FA
I77 (VDD VSS x3y3 net16 net12 net18 net19) TNOR_FA
I78 (VDD VSS net18 net14 bS6 aS7) TNOR_HA

//S7:
I79 (VDD VSS x7y0 x0y7 net13 net20 net21) TNOR_FA
I80 (VDD VSS x6y1 x1y6 net15 net22 net23) TNOR_FA
I81 (VDD VSS x5y2 x2y5 net17 net24 net25) TNOR_FA
I82 (VDD VSS x4y3 x3y4 net19 net26 net27) TNOR_FA
I83 (VDD VSS net20 net22 net24 net28 net29) TNOR_FA
I84 (VDD VSS net28 net26 bS7 aS8) TNOR_HA

//S8
I85 (VDD VSS x7y1 x1y7 net21 net30 net31) TNOR_FA
I86 (VDD VSS x6y2 x2y6 net23 net32 net33) TNOR_FA
I87 (VDD VSS x5y3 x3y5 net25 net34 net35) TNOR_FA
I88 (VDD VSS x4y4 net30 net27 net36 net37) TNOR_FA
I89 (VDD VSS net36 net32 net29 net38 net39) TNOR_FA
I90 (VDD VSS net38 net34 bS8 aS9) TNOR_HA

//S9
I92 (VDD VSS x7y2 x2y7 net31 net40 net41) TNOR_FA
I93 (VDD VSS x6y3 x3y6 net33 net42 net43) TNOR_FA
I94 (VDD VSS x5y4 x4y5 net35 net44 net45) TNOR_FA
I95 (VDD VSS net44 net42 net37 net46 net47) TNOR_FA
I96 (VDD VSS net46 net40 net39 bS9 aS10) TNOR_FA

//S10
I97 (VDD VSS x7y3 x3y7 net41 net48 net49) TNOR_FA
I98 (VDD VSS x6y4 x4y6 net43 net50 net51) TNOR_FA
I99 (VDD VSS x5y5 net45 net47 net52 net53) TNOR_FA
I100 (VDD VSS net52 net48 net50 bS10 aS11) TNOR_FA

//S11
```

```
I101 (VDD VSS x7y4 x4y7 net49 net54 net55) TNOR_FA
I102 (VDD VSS x6y5 x5y6 net51 net56 net57) TNOR_FA
I103 (VDD VSS net54 net56 net53 bS11 aS12) TNOR_FA

//S12
I104 (VDD VSS x7y5 x5y7 net55 net58 net59) TNOR_FA
I105 (VDD VSS net58 x6y6 net57 bS12 aS13) TNOR_FA

//S13
I106 (VDD VSS x7y6 x6y7 net59 bS13 aS14)TNOR_FA

//The fast Adder
I107 (VDD VSS x0y0 x1y0 aS2 aS3 aS4 aS5 aS6 aS7 aS8 aS9 aS10 aS11 aS12
aS13 aS14 VSS VSS x0y1 x1y1 bS3 bS4 bS5 bS6 bS7 bS8 bS9 bS10 bS11 bS12
bS13 x7y7 VSS VSS S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14
S15 Cout) TNOR_16bitMA
//Second zero is the beginning of B

ends TNOR_Wallace
// End of subcircuit definition.
```

16-bit 8-1 MUX

```
// Cell name: TNOR_MUX
// Function: 16-bit 8-1 Mux
// Inputs: A<15:0> B<15:0> C<15:0> D<15:0> E<15:0> F<15:0> G<15:0>
H<15:0> S0 S1       S2
// Outputs: out<15:0>
subckt TNOR_MUX VDD VSS A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13
A14 A15 B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12 B13 B14 B15 C0 C1 C2
C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15 D0 D1 D2 D3 D4 D5 D6 D7
D8 D9 D10 D11 D12 D13 D14 D15 E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10 E11
E12 E13 E14 E15 F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 F13 F14 F15
G0 G1 G2 G3 G4 G5 G6 G7 G8 G9 G10 G11 G12 G13 G14 G15 H0 H1 H2 H3 H4
H5 H6 H7 H8 H9 H10 H11 H12 H13 H14 H15 S0 S1 S2 out0 out1 out2 out3
out4 out5 out6 out7 out8 out9 out10 out11 out12 out13 out14 out15

I0 (VDD VSS A0 B0 C0 D0 E0 F0 G0 H0 S0 S1 S2 out0) TNOR_MUX8to1
I1 (VDD VSS A1 B1 C1 D1 E1 F1 G1 H1 S0 S1 S2 out1) TNOR_MUX8to1
I2 (VDD VSS A2 B2 C2 D2 E2 F2 G2 H2 S0 S1 S2 out2) TNOR_MUX8to1
I3 (VDD VSS A3 B3 C3 D3 E3 F3 G3 H3 S0 S1 S2 out3) TNOR_MUX8to1
I4 (VDD VSS A4 B4 C4 D4 E4 F4 G4 H4 S0 S1 S2 out4) TNOR_MUX8to1
I5 (VDD VSS A5 B5 C5 D5 E5 F5 G5 H5 S0 S1 S2 out5) TNOR_MUX8to1
I6 (VDD VSS A6 B6 C6 D6 E6 F6 G6 H6 S0 S1 S2 out6) TNOR_MUX8to1
I7 (VDD VSS A7 B7 C7 D7 E7 F7 G7 H7 S0 S1 S2 out7) TNOR_MUX8to1
I8 (VDD VSS A8 B8 C8 D8 E8 F8 G8 H8 S0 S1 S2 out8) TNOR_MUX8to1
I9 (VDD VSS A9 B9 C9 D9 E9 F9 G9 H9 S0 S1 S2 out9) TNOR_MUX8to1
I10 (VDD VSS A10 B10 C10 D10 E10 F10 G10 H10 S0 S1 S2 out10)
TNOR_MUX8to1
```

```
I11 (VDD VSS A11 B11 C11 D11 E11 F11 G11 H11 S0 S1 S2 out11)
TNOR_MUX8to1
I12 (VDD VSS A12 B12 C12 D12 E12 F12 G12 H12 S0 S1 S2 out12)
TNOR_MUX8to1
I13 (VDD VSS A13 B13 C13 D13 E13 F13 G13 H13 S0 S1 S2 out13)
TNOR_MUX8to1
I14 (VDD VSS A14 B14 C14 D14 E14 F14 G14 H14 S0 S1 S2 out14)
TNOR_MUX8to1
I15 (VDD VSS A15 B15 C15 D15 E15 F15 G15 H15 S0 S1 S2 out15)
TNOR_MUX8to1
ends TNOR_MUX
//End of subcircuit definition
```

ALU

```
// Cell Name: TNOR_ALU
// Function: 16 bit ALU, doesn't include input and output registers
// Inputs: Ain0 Ain1 Ain2 Ain3 Ain4 Ain5 Ain6 Ain7 Ain8 Ain9 Ain10
Ain11 Ain12 Ain13 Ain14 Ain15 Bin0 Bin1 Bin2 Bin3 Bin4 Bin5 Bin6 Bin7
Bin8 Bin9 Bin10 Bin11 Bin12 Bin13 Bin14 Bin15 con0 con1 con2 CLK
// Outputs: aluout0 aluout1 aluout2 aluout3 aluout4 aluout5 aluout6
aluout7 aluout8 aluout9 aluout10 aluout11 aluout12 aluout13 aluout14
aluout15 cout
subckt TNOR_ALU VDD VSS Ain0 Ain1 Ain2 Ain3 Ain4 Ain5 Ain6 Ain7 Ain8
Ain9 Ain10 Ain11 Ain12 Ain13 Ain14 Ain15 Bin0 Bin1 Bin2 Bin3 Bin4 Bin5
Bin6 Bin7 Bin8 Bin9 Bin10 Bin11 Bin12 Bin13 Bin14 Bin15 LRin con0 con1
con2 CLK aluout0 aluout1 aluout2 aluout3 aluout4 aluout5 aluout6
aluout7 aluout8 aluout9 aluout10 aluout11 aluout12 aluout13 aluout14
aluout15 cout

I0 (VDDA VSSA aluout0 aluout1 aluout2 aluout3 aluout4 aluout5 aluout6
aluout7 aluout8 aluout9 aluout10 aluout11 aluout12 aluout13 aluout14
aluout15 CLK muxA0 muxA1 muxA2 muxA3 muxA4 muxA5 muxA6 muxA7 muxA8
muxA9 muxA10 muxA11 muxA12 muxA13 muxA14 muxA15) TNOR_REG

I1 (VDDB VSSB Ain0 Ain1 Ain2 Ain3 Ain4 Ain5 Ain6 Ain7 Ain8 Ain9 Ain10
Ain11 Ain12 Ain13 Ain14 Ain15 con0 muxB0 muxB1 muxB2 muxB3 muxB4 muxB5
muxB6 muxB7 muxB8 muxB9 muxB10 muxB11 muxB12 muxB13 muxB14 muxB15)
TNOR_PASSA

I2 (VDDC VSSC Ain0 Ain1 Ain2 Ain3 Ain4 Ain5 Ain6 Ain7 Ain8 Ain9 Ain10
Ain11 Ain12 Ain13 Ain14 Ain15 Bin0 Bin1 Bin2 Bin3 Bin4 Bin5 Bin6 Bin7
Bin8 Bin9 Bin10 Bin11 Bin12 Bin13 Bin14 Bin15 con0 muxC0 muxC1 muxC2
muxC3 muxC4 muxC5 muxC6 muxC7 muxC8 muxC9 muxC10 muxC11 muxC12 muxC13
muxC14 muxC15 cout) TNOR_16bitMA

I3 (VDDE VSSE Ain0 Ain1 Ain2 Ain3 Ain4 Ain5 Ain6 Ain7 Ain8 Ain9 Ain10
Ain11 Ain12 Ain13 Ain14 Ain15 Bin0 Bin1 LRin muxE0 muxE1 muxE2 muxE3
muxE4 muxE5 muxE6 muxE7 muxE8 muxE9 muxE10 muxE11 muxE12 muxE13 muxE14
muxE15) TNOR_Shifter
```

```
I4 (VDDF VSSF Ain0 Ain1 Ain2 Ain3 Ain4 Ain5 Ain6 Ain7 Ain8 Ain9 Ain10
Ain11 Ain12 Ain13 Ain14 Ain15 Bin0 Bin1 Bin2 Bin3 Bin4 Bin5 Bin6 Bin7
Bin8 Bin9 Bin10 Bin11 Bin12 Bin13 Bin14 Bin15 muxF0 muxF1 muxF2 muxF3
muxF4 muxF5 muxF6 muxF7 muxF8 muxF9 muxF10 muxF11 muxF12 muxF13 muxF14
muxF15) TNOR_AND

I5 (VDDG VSSG Ain0 Ain1 Ain2 Ain3 Ain4 Ain5 Ain6 Ain7 Ain8 Ain9 Ain10
Ain11 Ain12 Ain13 Ain14 Ain15 Bin0 Bin1 Bin2 Bin3 Bin4 Bin5 Bin6 Bin7
Bin8 Bin9 Bin10 Bin11 Bin12 Bin13 Bin14 Bin15 muxG0 muxG1 muxG2 muxG3
muxG4 muxG5 muxG6 muxG7 muxG8 muxG9 muxG10 muxG11 muxG12 muxG13 muxG14
muxG15) TNOR_OR

I6 ( VDD VSS Ain0 Ain1 Ain2 Ain3 Ain4 Ain5 Ain6 Ain7 Bin0 Bin1 Bin2
Bin3 Bin4 Bin5 Bin6 Bin7 muxH0 muxH1 muxH2 muxH3 muxH4 muxH5 muxH6
muxH7 muxH8 muxH9 muxH10 muxH11 muxH12 muxH13 muxH14 muxH15 cout)
TNOR_Wallace

I7 (VDDM VSSM muxA0 muxA1 muxA2 muxA3 muxA4 muxA5 muxA6 muxA7 muxA8
muxA9 muxA10 muxA11 muxA12 muxA13 muxA14 muxA15 muxB0 muxB1 muxB2
muxB3 muxB4 muxB5 muxB6 muxB7 muxB8 muxB9 muxB10 muxB11 muxB12 muxB13
muxB14 muxB15 muxC0 muxC1 muxC2 muxC3 muxC4 muxC5 muxC6 muxC7 muxC8
muxC9 muxC10 muxC11 muxC12 muxC13 muxC14 muxC15 muxC0 muxC1 muxC2
muxC3 muxC4 muxC5 muxC6 muxC7 muxC8 muxC9 muxC10 muxC11 muxC12 muxC13
muxC14 muxC15 muxE0 muxE1 muxE2 muxE3 muxE4 muxE5 muxE6 muxE7 muxE8
muxE9 muxE10 muxE11 muxE12 muxE13 muxE14 muxE15 muxF0 muxF1 muxF2
muxF3 muxF4 muxF5 muxF6 muxF7 muxF8 muxF9 muxF10 muxF11 muxF12 muxF13
muxF14 muxF15 muxG0 muxG1 muxG2 muxG3 muxG4 muxG5 muxG6 muxG7 muxG8
muxG9 muxG10 muxG11 muxG12 muxG13 muxG14 muxG15 muxH0 muxH1 muxH2
muxH3 muxH4 muxH5 muxH6 muxH7 muxH8 muxH9 muxH10 muxH11 muxH12 muxH13
muxH14 muxH15 con0 con1 con2 aluout0 aluout1 aluout2 aluout3 aluout4
aluout5 aluout6 aluout7 aluout8 aluout9 aluout10 aluout11 aluout12
aluout13 aluout14 aluout15) TNOR_MUX

ends TNOR_ALU
//Ends subcircuit definition
```

ALU DSP Block with Top Level Connections (this acted as our Test Bench)

```
// Test Bench
// ALU with Top Level Connections (input and output registers)

V0 (vdd! 0) vsource dc=1.1 type=dc

I1 (vdd! 0 inA0 A0) InvBuffer
I2 (vdd! 0 inA1 A1) InvBuffer
I3 (vdd! 0 inA2 A2) InvBuffer
I4 (vdd! 0 inA3 A3) InvBuffer
I5 (vdd! 0 inA4 A4) InvBuffer
I6 (vdd! 0 inA5 A5) InvBuffer
```

```
I7 (vdd! 0 inA6 A6) InvBuffer
I8 (vdd! 0 inA7 A7) InvBuffer
I9 (vdd! 0 inA8 A8) InvBuffer
I10 (vdd! 0 inA9 A9) InvBuffer
I11 (vdd! 0 inA10 A10) InvBuffer
I12 (vdd! 0 inA11 A11) InvBuffer
I13 (vdd! 0 inA12 A12) InvBuffer
I14 (vdd! 0 inA13 A13) InvBuffer
I15 (vdd! 0 inA14 A14) InvBuffer
I16 (vdd! 0 inA15 A15) InvBuffer

I17 (vdd! 0 inB0 B0) InvBuffer
I18 (vdd! 0 inB1 B1) InvBuffer
I19 (vdd! 0 inB2 B2) InvBuffer
I20 (vdd! 0 inB3 B3) InvBuffer
I21 (vdd! 0 inB4 B4) InvBuffer
I22 (vdd! 0 inB5 B5) InvBuffer
I23 (vdd! 0 inB6 B6) InvBuffer
I24 (vdd! 0 inB7 B7) InvBuffer
I25 (vdd! 0 inB8 B8) InvBuffer
I26 (vdd! 0 inB9 B9) InvBuffer
I27 (vdd! 0 inB10 B10) InvBuffer
I28 (vdd! 0 inB11 B11) InvBuffer
I29 (vdd! 0 inB12 B12) InvBuffer
I30 (vdd! 0 inB13 B13) InvBuffer
I31 (vdd! 0 inB14 B14) InvBuffer
I32 (vdd! 0 inB15 B15) InvBuffer

I33 (vdd! 0 inLRin LRin) InvBuffer

I34 (vdd! 0 regcon0 CLK incon0) TNOR_REG1
I35 (vdd! 0 regcon1 CLK incon1) TNOR_REG1
I36 (vdd! 0 regcon2 CLK incon2) TNOR_REG1

I37 (vdd! 0 incon0 con0) InvBuffer
I38 (vdd! 0 incon1 con1) InvBuffer
I39 (vdd! 0 incon2 con2) InvBuffer

I40 (vdd! 0 inCLK CLK) InvCLKBuffer

I41 (vdd! 0 A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 CLK
Ain0 Ain1 Ain2 Ain3 Ain4 Ain5 Ain6 Ain7 Ain8 Ain9 Ain10 Ain11 Ain12
Ain13 Ain14 Ain15)    TNOR_REG

I42 (vdd! 0 B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12 B13 B14 B15 CLK
Bin0 Bin1 Bin2 Bin3 Bin4 Bin5 Bin6 Bin7 Bin8 Bin9 Bin10 Bin11 Bin12
Bin13 Bin14 Bin15) TNOR_REG

I43 (vdd! 0 Ain0 Ain1 Ain2 Ain3 Ain4 Ain5 Ain6 Ain7 Ain8 Ain9 Ain10
Ain11 Ain12 Ain13 Ain14 Ain15 Bin0 Bin1 Bin2 Bin3 Bin4 Bin5 Bin6 Bin7
```

Matthew Ridder,
Doug Etts,
Joseph Fackler

```
Bin8 Bin9 Bin10 Bin11 Bin12 Bin13 Bin14 Bin15 LRin con0 con1 con2 CLK
aluout0 aluout1 aluout2 aluout3 aluout4 aluout5 aluout6 aluout7
aluout8 aluout9 aluout10 aluout11 aluout12 aluout13 aluout14 aluout15
cout) TNOR_ALU

I44 (vdd! 0 aluout0 aluout1 aluout2 aluout3 aluout4 aluout5 aluout6
aluout7 aluout8 aluout9 aluout10 aluout11 aluout12 aluout13 aluout14
aluout15 CLK out0 out1 out2 out3 out4 out5 out6 out7 out8 out9 out10
out11 out12 out13 out14 out15) TNOR_REG

I45 (vdd! 0 out0 0) TNOR_Inverter wp=720n wn=360n
I46 (vdd! 0 out1 0) TNOR_Inverter wp=720n wn=360n
I47 (vdd! 0 out2 0) TNOR_Inverter wp=720n wn=360n
I48 (vdd! 0 out3 0) TNOR_Inverter wp=720n wn=360n
I49 (vdd! 0 out4 0) TNOR_Inverter wp=720n wn=360n
I50 (vdd! 0 out5 0) TNOR_Inverter wp=720n wn=360n
I51 (vdd! 0 out6 0) TNOR_Inverter wp=720n wn=360n
I52 (vdd! 0 out7 0) TNOR_Inverter wp=720n wn=360n
I53 (vdd! 0 out8 0) TNOR_Inverter wp=720n wn=360n
I54 (vdd! 0 out9 0) TNOR_Inverter wp=720n wn=360n
I55 (vdd! 0 out10 0) TNOR_Inverter wp=720n wn=360n
I56 (vdd! 0 out11 0) TNOR_Inverter wp=720n wn=360n
I57 (vdd! 0 out12 0) TNOR_Inverter wp=720n wn=360n
I58 (vdd! 0 out13 0) TNOR_Inverter wp=720n wn=360n
I59 (vdd! 0 out14 0) TNOR_Inverter wp=720n wn=360n
I60 (vdd! 0 out15 0) TNOR_Inverter wp=720n wn=360n
I61 (vdd! 0 cout 0) TNOR_Inverter wp=720n wn=360n
```